ITEA Journal 2008; 29: 380–387
Copyright © 2008 by the International Test and Evaluation Association

Testing the Intelligence of Unmanned Autonomous Systems

Miles Thompson
Trideum Corporation,
Aberdeen Proving Ground, Maryland

There is a common misconception in the testing industry that all unmanned autonomous systems can be tested using methodologies developed to test manned systems. A tester can incorrectly state that driving a manned vehicle is conceptually identical to driving an unmanned vehicle; the only difference lies in the position of the operator. In actuality, the main difference lies in the unmanned autonomous system's role in the decision process. Because testers cannot make assumptions about the decision process within an unmanned autonomous system, there is a need for a methodology that completely tests this decision process without biasing the system into a default "human" solution. This article presents a comprehensive methodology for testing the intelligence of an unmanned autonomous system within a given set of requirements.

Key words: Autonomous systems; decision process; intelligence; unmanned autonomous system.

There is a common misconception in the testing industry that all unmanned systems can be tested using methodologies developed to test manned systems. The operation of some unmanned systems is conceptually identical to operating manned systems except for the position of operator. However, other unmanned systems differ from manned systems not only in the position of the operator, but also in their role in the decision process. In the robotics industry, these unmanned systems are referred to as intelligent or autonomous.

The terms unmanned ground vehicle (UGV) or unmanned air vehicle (UAV) are much more common within the military, but these terms are not descriptive enough for this methodology. It is possible to have a UGV or UAV that has no choice in the decision process; radio controlled cars and planes are good examples. Instead, an intelligent unmanned system is a system that takes data, perceives information, possibly compares the information against knowledge, and makes a decision based upon this information. In this way, a teleoperated, unmanned system could be considered intelligent only if the system being tested included the operator or the system makes some decisions without the operator. The term autonomous is also used interchangeably with intelligent, giving rise to the name unmanned autonomous system (UAS). For the purposes of this article, UAS describes an unmanned system that makes decisions based on gathered information.

Because testers should not make assumptions about the decision process within a UAS, there is a need for a methodology that completely tests this decision process without biasing the system into a default "human" solution. In this article, the reader will discover a new, comprehensive methodology for testing the intelligence of a UAS while testing explicitly to a given set of requirements.

Unmanned autonomous system testing (UAST)

To test any system effectively, a tester must be given three things: the system to be tested, the documentation associated with operation and maintenance of the system to be tested, and the requirements against which the system will be tested. An intelligent unmanned system is no different. For testing the intelligence of a UAS, this does not necessarily mean an entire vehicle. The system under test could instead be the removed computer from a UAS as a preliminary subsystem test in simulation. In fact, testing the decision making capability of the computer before its being placed within a vehicle is not only cheaper, it is much safer. The point to remember here is that not all UAS testing requires the entire system and that these

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2008**	2. REPORT TYPE	3. DATES COVERED **00-00-2008 to 00-00-2008**
4. TITLE AND SUBTITLE **Testing the Intelligence of Unmanned Autonomous Systems**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Trideum Corporation,1202-B Technology Drive,Aberdeen,MD,35805-5906**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

selected tests can be run in preparation for tests of the entire system. At the same time, no testing process is complete without at some point testing the entire system.

The documentation for the operation and maintenance of the UAS is critical to developing representative tests. This documentation gives the testers insight into how the developers envision the end-users will operate and maintain the UAS. Additionally, the documentation associated with a UAS makes testers aware of safety concerns through extensive hazard analysis. This analysis should show that the UAS would fail gracefully and without incident.

Finally is a list of requirements against which the system can be tested. Requirements are specifications that a test article must meet or exceed. Metrics, which are quantifiable data that pertain to a specific requirement, are used to assess satisfaction of requirements. Given just these three things, testers need to create an effective and efficient test schedule. The process to design this test schedule is the methodology all UAS testers should use for consistent evaluation.

Test to requirements

The requirements are the best place to start designing a methodology because they are the basis of all testing. Requirements are specifications that a test article must meet or exceed. Many requirements dictate the missions and environments the test article will encounter in the field and are required for creating the critical test matrix. The critical test matrix is an array of test scenarios with environments along one dimension and missions along another. The difference between the critical test matrix and any other test matrix is that these are the bare minimum test scenarios necessary to observe the behavior of a UAS and to record enough data for later virtual environment verification. Other requirements can fit into classes of requirements that are important for parallel testing. Parallel testing is a strategy that tests more than one requirement per test. Testers then use classes of requirements to identify specific measurement and evaluation tools necessary for the test. Each class represents a type of sub-test within each test scenario in the critical test matrix, and the set of all the sub-tests creates a complete test. In order to show the process in action, here is an example based on typical UAS requirements.

The UGV must be able to create a suitable path plan of movement to a goal waypoint and execute the plan under the following conditions:

1. Dynamic environment with moving obstacles possibly up to 65 MPH;
2. Must recognize all impassable obstacles at time 0.75 seconds before predicted impact at current speed along linear path to obstacle and avoid collision with any obstacle;
3. Must recognize all pedestrian human obstacles at time 1.5 seconds before predicted impact at current speed along linear path to obstacle and avoid collision while adjusting speed appropriately;
4. Maintain these top speeds: Off-Road Flat—40 MPH, Highway—55 MPH, Off-Road Complex/Forested—25 MPH, Human Within Possible Collision Trajectory—15 MPH;
5. Maintain mobility for 95 percent of mission duration up to 4 hours;
6. Allow an operator to take control of the vehicle for teleoperation at any point in the mission and later relinquish control back to the UGV;
7. Allow an operator to send goal waypoints to the UGV as a queue or interrupt a current waypoint with a new one.

Keep in mind, these requirements for the UGV are only pertaining to the autonomous or intelligent behavior and control and therefore qualify as UAS. There are additional requirements for mobility, sensing, and reliability that would use traditional manned vehicle test methodologies. In order to create a test matrix, testers need to extract a list of missions and environments from the requirements documentation. A paired mission and environment constitutes a scenario, and the test matrix will consist of all combinations of missions and environments, also known as all the scenarios. In this example, there is only one mission and that is to get to the end waypoint. From the requirements, we can also determine there are only four environments specified: highway, off-road flatlands, forest, and off-road complex. The testers also know that all the environments are dynamic with moving obstacles up to 65 MPH.

With the goals and environments clearly defined, the scenarios for critical test are set. Now the testers need to know exactly which metrics to run during each of these scenarios. The classes for sub-tests include, but are not limited to, object recognition, collision avoidance, operator control, path planning, time limit, and mobility control. Within object recognition, there are two sub-tests: human recognition and other impassable obstacle recognition. It can be inferred from the requirements that all environments have the possibility of human pedestrians as obstacles and therefore should be tested in every scenario. Collision avoidance only has one requirement; that is to avoid all impassable obstacles. Operator control has two different sub-tests: teleoperation and waypoint directive. The path planning class has an inherent sub-test that

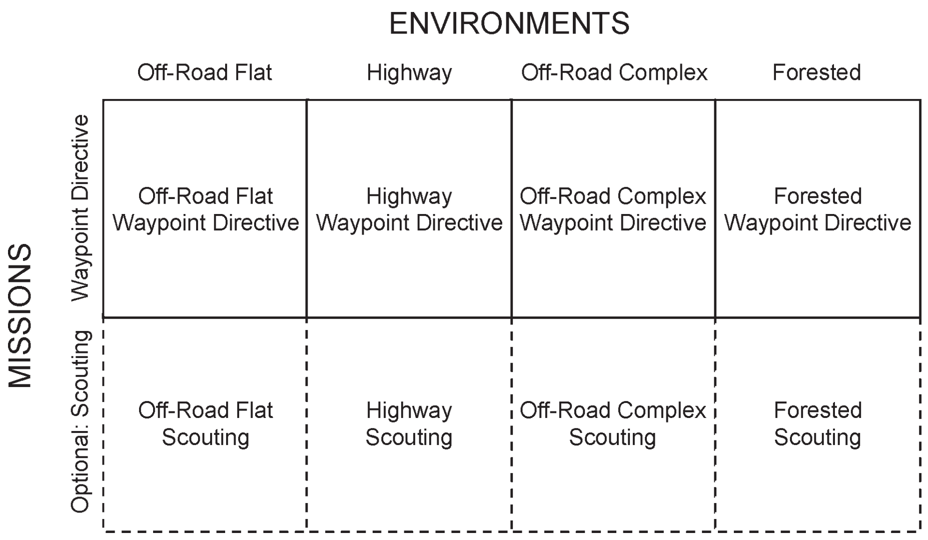

Figure 1. An example of this methodology creating a Critical Test Matrix

can be considered a pass if the UAS meets the collision avoidance and waypoint mission requirements. In the time limit class, the only time requirement is for the UAS to maintain mobility for 4 hours during mission. Mobility is an interesting class because there are two distinct sub-tests. The first is the obvious, maintaining mobility 95 percent of the time during the first 4 hours. The second is making sure the UAS does not exceed its maximum speed requirements in its given environments.

For this simple example, we end up with four mandatory tests and four optional tests *(Figure 1)*. The test matrix that is formed from this process is the critical test matrix because it contains all the necessary sub-tests to ensure that every requirement and every test scenario is tested at least once. During each of these tests, every identified sub-test will occur and the performance of the UAS will be measured with related metrics defined by each sub-test's requirement class *(Figure 2)*. Additionally, it is possible to have requirements be mission or environment dependent or even exclusive. In these cases, those sub-tests are only run when the scenario has the necessary mission or environment. Unfortunately, this amount of testing is not necessarily complete testing, because while the process generated the necessary scenarios, there was no iteration through the various parameters of each scenario. For example, if the sun had been 20 degrees lower in the sky from the zenith position, the UAS may have performed differently.

Complete testing—the critical and the unique test matrix

The critical test matrix is a good start to complete testing because it can give testers quick analysis of a UAS function in typical missions and environments. However, because it does not incorporate a scientific step through the scenario and mission parameter values, it can hardly be considered complete. Traditional Design of Experiments would suggest a full factorial test design or possibly a Plackett-Burman test design for creating a complete multivariate test matrix (NIST 2008). The full factorial test design just ends up being impractical. For example, a UAS that is placed in 10 scenarios with an average of 30 parameters each and 20 discreet values for each parameter has a test matrix with an order of magnitude of 10^{40} tests. The Plackett-Burman design would be better, even if only to reveal the most influential parameters, but this test design grows exponentially for each value over two for each parameter, bringing it to an unmanageable size very quickly. A new method of Design of Experiments is required to test a UAS comprehensively while at the same time limiting the size of the test matrix or developing ways to increase the speed and decrease the cost of testing.

Parameter effect propagation

The Robotic Intelligence Evaluation Program (RIEP) at Aberdeen Test Center (ATC) has designed a methodology to both comprehensively test the intelligence of a UAS and limit the size and complexity of the tests. Through parameter effect propagation, testers can limit the size of the test matrix by running only those tests that provide a unique situation to a UAS. Parameter effect propagation is the process of recognizing all the individual sets of parameter values (i.e., all the possible scenarios) and estimating the changes to the perception and interaction factors of the UAS. Factors are test article based variables that alter

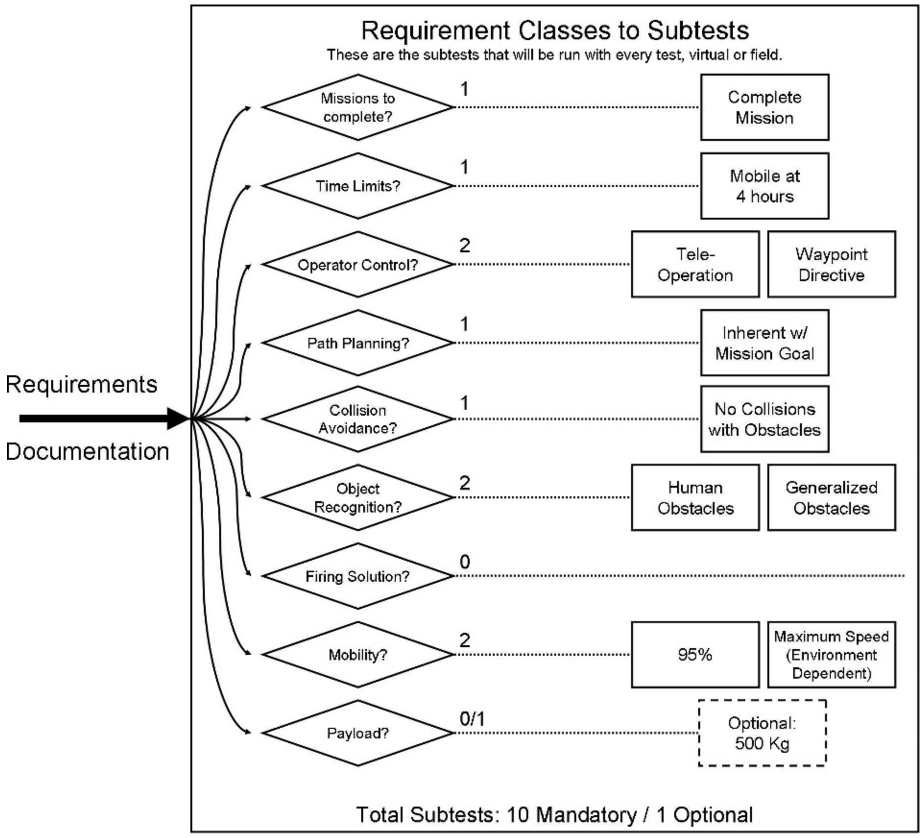

Figure 2. The complete set of sub-tests for every test scenario

the function of the system (e.g., coefficient of friction, camera visibility, battery power), and many are determined by the scenario parameters *(Figure 3)*. For example, an unlit video camera would return much less information during a night scenario compared to a day scenario. At the same time, a night scenario might return the same information as a broken camera scenario, and the UAS will treat them as the same.

The Parameter effect propagation process requires an intimate understanding of the subsystems of a UAS. Each subsystem of the UAS, including redundant subsystems, contributes to the overall collection of factors. For example, a UAS with eight similar sensors will have at least eight factors that are sensor specific. The data from each of these sensors are a response that could be altered by the sensor factors. A change in the response of any subsystem could alter the capability of the system as a whole.

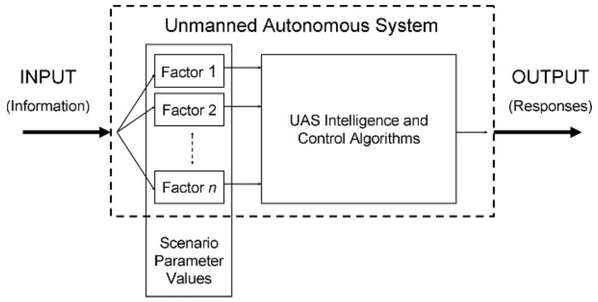
Unmanned Autonomous System
INPUT
(Information)
Factor 1
Factor 2
Factor n
UAS Intelligence and Control Algorithms
OUTPUT
(Responses)
Scenario Parameter Values

Figure 3. The influence a parameter value has on the system can be better represented by the effect the value has on a system factor

Before testers can determine how the parameters of a scenario will uniquely affect a UAS, they need to establish the parameters. Using the list of scenarios developed from the requirements documentation, testers can produce a list of scenario parameters for each. The list of scenario parameters is determined by examining the variables in a scenario's mission along with all the ways the environment can interact with the system. Initially, this method might appear to yield an enormous amount of parameters, but the environment can only interact with the system in a few ways. In addition to the physical interaction and electromagnetic interference between the environment and the UAS, the only other inputs into the system are determined by the sensor subsystems. In other words, just like a human, a UAS can only base decisions on what it detects.

With a complete list of parameters, the testers next need to know which values to use for each. Discreet

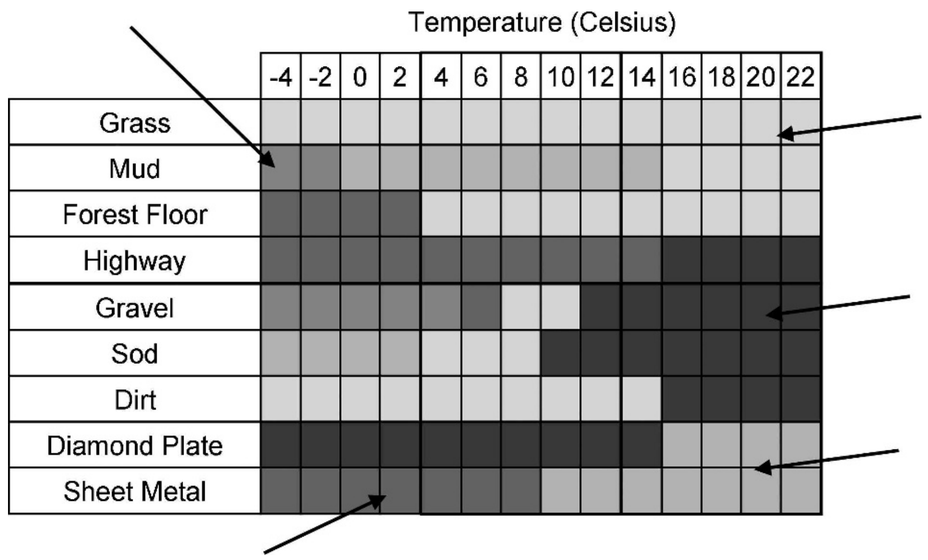

Figure 4. An example of how parameter effect propagation leads to unique testing regions. These unique testing regions give testers the necessary tests to construct a unique test matrix

parameters are mostly straightforward but if the parameter is continuous or there is a multitude of discreet values, there is an approach for selecting the proper values. When traditionally testing a system, testers tend to choose values at the 95 and 99 percent extremes on the probability distribution function of a parameter. With a system containing so many parameters, testing only the extremes would leave possible destructive daily parameter interactions untested. However, selecting parameter values along a probability distribution function at a given percentage step size not only allows the tester to select the extremes but also more of the most common values. The percentage step size is determined by running a sensitivity analysis on the UAS factors by this particular parameter.

Finding the effects on the UAS factors is the next step. Effects on subsystem factors can be predicted through analytical calculations and then verified through field testing or through empirical experimentation. Testers should not be concerned with the multitude of parameter value combinations, but rather concerned with the sets of factor effects derived from the parameter permutations. Running 100 different tests that induce the same factor effect set in the system will not tell the tester nearly as much as running 100 different tests where each of which induces a unique factor effect set. A test team should step through each parameter value for each scenario and calculate a corresponding factor effect set. It is important to take the set of parameter values that define a scenario together so that the factor effect set can accurately represent interaction between multiple parameters. For example, smoke and lighting can each affect optical sensors, but the combination of different values of these two parameters can produce drastic effects on a system. Further, without lighting at all, it makes no difference to the optical sensor on a UAS how much smoke is in its path because it may not be able to "see" at all. When one parameter's effect makes another's effect on a factor inconsequential, it is called parameter dominance. If all the effects of all the values of one parameter dominate all the effects of all the values of another parameter, there is no reason to have the second parameter and it can be removed from the scenario.

Unique test matrix

In order to produce the unique test matrix, the testers need to have produced a factor effect set for each parameter value set. Many of these factor effect sets will be indistinguishable from one another given the precision used by the decision making process of the UAS. In order to limit the size of the test matrix while still representing a significant portion of the scenarios a UAS would encounter in the field, the testers remove extra parameter value sets that correspond to indistinguishable factor effect sets. In *Figure 4*, there is a simple example of this process where two parameters, terrain material and temperature, both affect a factor on the UAS—traction with the ground. The different shaded regions each represent a different value for traction and are also known as unique testing regions. To the system being tested, each similarly shaded case would be identical because the system is unable to realize a difference between them. If testers were to run every case, there would be 126 tests just for traction alone. Instead, choosing one case from each of the unique testing regions leaves testers with five tests and with a significant portion of the information they would have had running all 126 tests.

The resulting test matrix might be many orders of magnitude less than the original full factorial test design. This test matrix is called the unique test matrix because each test to be run should propose a unique problem to the UAS resulting in the greatest probability of inducing emergent and possibly unwanted behavior. A UAS with 12 distinct subsystems may only end up with a few hundred unique factor effect sets for each scenario. For the same 10 scenarios from the example before, the unique test matrix could have an order of magnitude of 10^3 test elements. Although 1,000 tests still sounds like a lot, testers can use simulation software to run virtual tests and identify possibly problematic field tests beforehand. In addition to the unique test matrix, the critical test matrix should be run in a virtual environment as well for verification of the virtual environment. Although executing the virtual tests depend on many conditions, such as the length of the scenario, the devoted processing power, and the discreet time interval in the test, 1,000 virtual tests could take as little as 1 day.

Necessities for virtually testing an UAS

Running an entire unique test matrix in the field is both temporally and financially impractical. Realistically, a unique test matrix will have anywhere from 1,000 to 1,000,000 tests for a complicated UAS, and these are only for testing the intelligent aspects of the system. To run 1,000,000 tests in the field would take 547 years at five tests a day 365 days a year. Even 1,000 tests would take the better half of a year at this same rate. In addition to the temporal and financial issues of running field tests, safety is becoming a much larger issue. Virtual testing allows a system to be preliminarily tested without the risk of bodily harm or damage to the test article. Virtual testing must become a standard complement to field-testing UASs if the testing community is ever going to be able to test an intelligent UAS safely and comprehensively. That being said, what are the necessary pieces to virtual testing?

Virtual environment resolution

To conduct a virtual test, a tester must have a virtual environment in which to run the test. Depending on the UAS, these environments could include ground, air, underwater, surface, space, or any combination of the above. The important thing is that the virtual environment is identical to the real environment in which the UAS would be operating. Identical sounds difficult, but it is really a relative term. If a UAS has no way to detect its environment other than touch, then what does it matter if the environment is all one color? If all the physical obstacles and surfaces in the real environment match the physical obstacles and surfaces in the virtual environment, this UAS would not know the difference and the environments are identical.

This leads to the point that the virtual environments must be interaction based. The term interaction is used because it is not enough to have excellent sensor models if the system cannot take action in the virtual environment. Unmanned ground systems can be mobile but very few actually sense terrain properties such as a friction, roughness, and material, which are all important from an environment perspective. Using lean software development is also very important when designing virtual environments for specific systems. Creating complex acoustic environments is a waste of time if the UAS does not possess any acoustic equipment. To create an effective virtual environment, testers need to study the UAS and understand not only what information it gathers, but also what environmental information is necessary for all interactions. For example, if a UAS has a range detecting LADAR that gathers ranges to millimeter precision, that data may be converted to a lower resolution for path planning. The virtual environment needs to be able to return the more precise data that the UAS will gather in the field to make sure that the UAS processes the information correctly.

Communications and control

From the earlier discussion, a UAS gets its perceived information from two sources: sensors and communications. While communications could be considered a form of sensor, it has its own special place in testing because the commands sent over communications are not determined by the environment the UAS is in. Commands may be sent at any time during a mission and may be as brief as a requesting status update to as verbose as an operator taking over the controls in teleoperation mode. Any commands that the UAS could receive in the field need to be accurately and completely represented in virtual testing. Some UASs even act as communication repeaters and need to be able to send communications on to other agents. This capability must also be modeled accurately within a virtual environment.

Expanding communications to operator control is another important aspect. A virtual environment must be able to interface with the command and control devices that will be used by the operator in the field and be able to update them with the information the operator needs for making decisions. Even autonomous vehicles that do not have a teleoperation mode require an operator to issue high order commands. A virtual environment that encapsulates these operator devices can also be used for operator workload tests without the risks of damaging expensive equipment.

Networking issues

If the virtual environment already needs to support operator equipment, then it might as well be distributed and scalable. This way, multiple UASs can take part in the same test as needed and operators can be stationed at different locations. The virtual test can be scaled up and interconnected with multiple test centers for a distributed test across many assets with many entities.

Aside from these preferences for complex virtual tests, latency is a major issue with networked virtual tests. Tests run in real time across a network introduce outside latency that must be minimized or accounted for. Tests run in virtual time with a higher rate of time passage would enable faster testing tests but follow the same latency issues. Unfortunately, using a faster (or even slower) virtual time is no longer testing the UAS in a realistic environment. Most UASs are already maxing out the computational load in real time and would gain a computational advantage in a slower-than-real-time test and have a disadvantage in a faster-than-real-time test.

Hardware-in-the-loop and vehicle-in-the-loop

Hardware-in-the-loop is a necessity of any virtual test environment. To make sure the UAS is tested on its own hardware, it needs to be able to directly connect to the virtual environment. There are a few ideas on how to do this. The least invasive way is to have a connection from the virtual environment to all the UAS's inputs from sensors and communications. In other words, take the UAS's computer out of the remainder of the system and connect the virtual sensors and communications in the exact same ports that the real ones use. This requires very intimate knowledge of the raw data each sensor produces and knowledge of the connection types. Another approach is to connect directly to the system bus. In order for this to work, messages coming off the sensors need to be transformed to represent what the virtual sensors are seeing and not what the real sensors see.

Vehicle-in-the-loop is a different and interesting new field. Instead of connecting the computer to the environment and seeing how it would perform in the virtual environment, testers are now able to leave the entire vehicle connected. This means a command to drive forward would not only move the virtual vehicle, but also the real vehicle. Already used for ground vehicles, the technology requires the UAS to be dynamically supported on what is known as a Roadway Simulator such as one at the ATC. The advantage of vehicle-in-the-loop testing is that it can be used to determine the effectiveness of the UAS control algorithms in addition to the decision process in a perfectly safe and contained environment where no one can get hurt and the UAS cannot be damaged.

A general procedure for testing a UAS is progressive. It is recommended that the platform is testing for structural integrity and controllability first. Along with these tests, it is important to test the internal communications and the ability of components to interact within the system. Next is hardware-in-the-loop intelligence testing followed by the vehicle-in-the-loop testing. Finally, actual field tests are necessary to verify and validate the results from the previous tests.

Verification and validation

"*Validation* is the process of determining the extent to which the models and simulations accurately represent the real world from the perspective of the intended use of the models and simulations" (ATEC 2007).

During a pre-test meeting, the testers, UAS developers, and end-users should have come to a decision regarding the validity of the testers' approach, execution, and analysis of the UAS and prepared a written statement stating this fact. If so, the validation aspect is complete.

"*Verification* is the process of determining if a model and/or simulation (M&S) accurately represents the developer's conceptual description and specifications and meets the needs stated in the requirements document …" (ATEC 2007).

To verify the credibility of the virtual tests as a representation of the real world, testers must compare the system's performance in the virtual test with the same system's performance in the related field test. However, before testers can compare performance between field and virtual tests, the environmental parameters must be compared to ensure that the virtual environment is representative of the field environment in every significant way. As testers of robotic intelligences, the most important metric for verification is replication of the decision process in the virtual and field tests. If, in a virtual test, a UAS successfully completes a mission but does not complete it in the field test, there is some environmental stimulus missing from the virtual test that is present in the field test or vice versa. Software systems are deterministic and will generate the same output for the same input so testers know there is an incorrect input in the virtual test. This

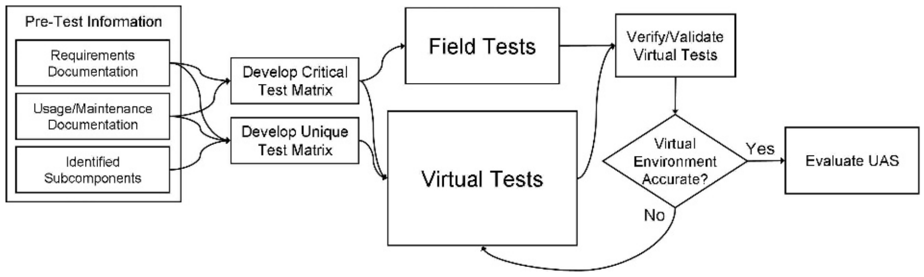
Pre-Test Information
Requirements Documentation
Usage/Maintenance Documentation
Identified Subcomponents
Develop Critical Test Matrix
Develop Unique Test Matrix
Field Tests
Virtual Tests
Verify/Validate Virtual Tests
Virtual Environment Accurate?
Yes
No
Evaluate UAS

Figure 5. The test design process for testing an unmanned autonomous system's decision process or intelligence

is even true for probabilistic algorithms, as the output should be measured as the probability distribution for actions and not necessarily the action taken.

Testers should examine the data from all replicated tests to make sure the UAS decisions are identical for each pair of tests. Recall that the replicated tests are those from the critical test matrix because they were run in both the field and virtual environments. Comparing the UAS logs from the virtual and field tests is vital to this process. There is no definitive way to understand a UAS decision process without acquiring the logs from that system. If there are any discrepancies, an iterative process is used to modify the simulation software so that it more accurately represents the stimuli the UAS acquires with its sensors. The ATC RIEP recommends that all simulation software be tested in advance for fidelity with sensor models and compared with a base test UAS or sensor capture system in the field.

Conclusion

Ultimately testers are left with a linear flowchart for testing a UAS. As can be seen in *Figure 5*, given the right starting information a test team can develop a very sophisticated test regimen for any UAS. One last reminder: the methodology discussed in this article is for testing the intelligent aspects of a UAS and is not a complete reference. For a complete testing of any system, testers need to incorporate traditional system testing to ensure that the physical and psychological aspects are tested as well. ❑

MILES THOMPSON is a Robotics Test Engineer for the Trideum Corporation based in Huntsville, Alabama. A 2006 graduate of Carnegie Mellon University, his work at Aberdeen Test Center is largely centered on the fields of robotics and mathematical models of systems. He is currently on the Robotic Intelligence Evaluation Program (RIEP) at Aberdeen Test Center. The goal of the program is to provide the U.S. Army, and the rest of the testing industry, methodologies for testing the intelligent systems. Because these methodologies rely heavily on simulation testing, the program intends to release tools for managing these simulations and collecting and analyzing data. E-mail: miles.thompson@atc.army.mil

References

ATEC. 2007. "Test and Evaluation Modeling and Simulation Verification, Validation, and Accreditation Methodology." ATEC Pamphlet 73-2113. April 2007. Alexandria, VA: US Army test and Evaluation Command.

NIST (National Institute of Standards and Technology). 2008. NIST/SEMATECH e-Handbook of Statistical Methods, Gaithersburg, MD: NIST. Available at http://www.itl.nist.gov/div898/handbook/. Accessed January 9, 2008.